(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 918 275 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
26.05.1999 Bulletin 1999/21

(51) Int. Cl.[6]: **G06F 1/00**

(21) Application number: **98111462.2**

(22) Date of filing: **22.06.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **13.11.1997 US 970064**

(71) Applicant:
Hewlett-Packard Company
Palo Alto, California 94304 (US)

(72) Inventors:
- **McCollom, William Girard**
  Fort Collins, CO 80525 (US)
- **Garcia, Julio Cesar**
  Fort Collins, CO 80525 (US)
- **Smith, Darren Drew**
  Fort Collins, CO 80526 (US)

(74) Representative:
Schoppe, Fritz, Dipl.-Ing.
Schoppe & Zimmermann
Patentanwälte
Postfach 71 08 67
81458 München (DE)

(54) A method of securing software configuration parameters with digital signatures

(57) A system and method for enforcing configuration parameters and detecting tampering of configuration files used by a software application. An enforced configuration packet (ECP) file generator (110) generates an enforced configuration packet (ECP) file (106) from a configuration parameter description file (108) containing a set of configuration parameters (⟨ID, VALUE⟩). The ECP file (106) includes a set of enforced configuration packets (⟨ID, VALUE, FINGERPRINT⟩), which each include one of the configuration parameters from the ECP description file (108) and a corresponding configuration parameter fingerprint (⟨FINGERPRINT⟩) unique to that particular configuration parameter. At startup of the software application (102), an ECP file reader (104) validates the ECP file (106) and each of the enforced configuration packets contained in the ECP file. Validation is achieved by regenerating the configuration parameter fingerprint of each configuration parameter and comparing the regenerated fingerprint to the fingerprint contained in the enforced configuration packet. If any of the fingerprints in the enforced configuration packets do not match their regenerated fingerprint, the ECP file reader (104) indicates that the configuration parameter fingerprint is not valid. If all of the configuration parameter fingerprints match up to their regenerated fingerprints, and the ECP file (106) itself is determined to be valid, the configuration parameters encoded in the ECP file (106) are used by the software application (102) to set up its configuration.

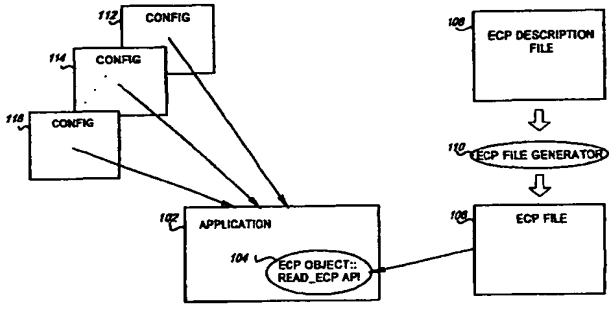

FIG. 1

EP 0 918 275 A2

Description

Field of the Invention

[0001] The present invention relates generally to the field of software security, and more particularly to a system and method for securing software configuration parameters with digital signatures.

Background of the Invention

[0002] Present day computer software is typically generated by a software manufacturer and shipped as a set of files. These files may include an installation program, one or more executable files, a set of library or data files, and configuration files. Typically, the end user of the software runs the installation program, which copies the executable files and libraries into appropriate directories, and configures the software according to configuration parameters contained in the configuration files. Often, the software configuration may be customized by either an intermediate customer (i.e, an Original Equipment Manufacturer (OEM) or Value Added Reseller (VAL) selling to end users) or by the software manufacturer working with the intermediate customer to meet the customers' particular needs. Software configuration customization is typically achieved today by providing a set of user customizable configuration files which may be modified by the intermediate customer, or by the manufacturer according to the customer's specifications, to configure the software to provide specific software options for the end user. An example of a configuration file may be a menu registration file that contains configuration parameters that describe the types, layout and functionality of menu bars displayed in conjunction with, and utilized by, the software application. The menu registration file is typically in text format, where each different menu bar is specified by including an entry containing a menu identifier and textual description of the contents or functionality of the menu. The contents of the menu registration file are used to generate the menu bar. Accordingly, if a customer requires an additional menu bar, the menu registration file may be modified to include an entry containing the menu identifier and information from which to generate the new menu bar or additional files from which to add new functionality to the menus.

[0003] The above described mechanism for providing customer-specific software configurations is both efficient and convenient in terms of development and maintainability for software manufacturers, intermediate customers, and end users. Specifically, the original executable software may be modified, and the new revision may be shipped and installed by end users without requiring a special version of software to be created for each customer specific configuration. In addition, an intermediate customer may add, remove, and modify configurable features that it wants to ship to its end users simply by modifying the customer specific configuration files, rather than requiring a special version of software to be created for its new customer specific configuration.

[0004] One aspect of providing customizable configuration files that has heretofore been unaddressed is the vulnerability of the customizable configuration files to end-user tampering. Specifically, software manufacturers provide non-secure customizable configuration files to allow an intermediate customer to edit these configuration files, as for example to Specify particular menu items and software functionality. However, these same customizable configuration files containing customized configuration parameters may also be shipped to the end user in non-secure form. It may be undesirable to certain customers to allow end users to have access to its customer specific configuration files. Accordingly, a need exists for a mechanism which allows a software manufacturer and a customer to customize the software configuration and then to secure the configuration parameters to disallow tampering by unauthorized users.

Summary of the Invention

[0005] A novel system method of enforcing a software configuration parameters in a software application for end users and for ensuring that configuration parameters and configuration files have not been tampered with is presented herein. In accordance with the system of the invention, an enforced configuration packet (ECP) file generator is provided to generate an enforced configuration packet (ECP) file from a configuration parameter description file, which contains a set of desired configuration parameters. The ECP file generator generates a fingerprint for each of the configuration parameters contained in the ECP description file and packages them each into a respective enforced configuration packet which is output to the ECP file. Each enforced configuration packet includes a configuration parameter and a configuration parameter fingerprint. Further in accordance with the system of the invention, an ECP file reader is provided which validates the enforced configuration packet in the ECP file and reads the configuration parameters if the ECP file and each of the ECP configuration parameters are valid. If any of the ECP configuration parameters or the file itself is invalid, the ECP file reader indicates that the configuration parameter is not valid.

[0006] In a preferred embodiment, a configuration parameter may include a name of a file or a string of filenames whose contents are included when generating the configuration parameter fingerprint. Also in a preferred embodiment, a configuration parameter may be overridable by a corresponding configuration parameter encoded in another ECP file such that as long as at least one of the enforced configuration packets in one of the ECP files includes a valid fingerprint of the configu-

ration parameter, the ECP file reader interprets it as valid.

[0007] A key parameter may be used to generate a key fingerprint to allow only authorized users to run the application. Typically, the key parameter will be an intermediate customer key, which may be included for example in a software registration code shipped with the software. Without a valid key parameter, the ECP file reader does not return the configuration parameters, and accordingly, the software application will not allow the user to run the application.

[0008] An additional security feature is the inclusion of an end-of-file (EOF) fingerprint appended to the end of the ECP file. The EOF fingerprint is generated on the entire contents of the ECP file to ensure that none of the contents of the ECP file itself have been tampered with.

[0009] In accordance with the invention, the method includes steps for generating a configuration parameter description file comprising a configuration parameter, generating an enforced configuration packet (ECP) file comprising the configuration parameter and a configuration parameter fingerprint, and providing the software application with means for validating the enforced configuration packet in the ECP file. Preferably, the software application includes steps for utilizing the configuration parameter if the enforced configuration packet is valid, and disallowing use of the configuration parameter if the enforced configuration packet is not valid. To ensure that the contents of files named in the configuration parameter, the configuration parameter fingerprint may also be generated on contents of the file or files named in the configuration parameter.

[0010] To ensure that an authorized user is running the application, the method of the invention may also include a step for generating a key fingerprint on a key parameter and appending the key fingerprint to the contents of the ECP file, wherein the key parameter must be known to validate the ECP file. The method may also include a step for requiring the means for validating the enforced configuration packet in the ECP file to perform the steps of generating a regenerated key fingerprint on the key parameter; comparing the regenerated key fingerprint to the key fingerprint in the ECP file; and returning an error code if the regenerated key fingerprint and the key fingerprint do not match.

[0011] To ensure that the ECP file itself has not been tampered with, the method of the invention may also include steps for generating an end-of-file (EOF) fingerprint on entire contents of the ECP file after the enforced configuration packets are generated and appending the EOF fingerprint to the contents of the ECP file; and requiring the means for validating the enforced configuration packet in the ECP file to perform the steps of generating a regenerated end-of-file (EOF) fingerprint on the contents of the ECP file; comparing the regenerated EOF fingerprint to the EOF fingerprint in the ECP file; and returning an error code if the regenerated EOF fingerprint and the EOF fingerprint do not match.

Brief Descriptions of the Drawings

[0012] The objects and advantages of the invention will become more apparent and more readily appreciated from the following detailed description of the presently preferred exemplary embodiment of the invention taken in conjunction with the accompanying drawing, of which:

FIG. 1 is a block diagram of a system in which the invention may operate;
FIG. 2 is an example format of an ECP description file;
FIG. 3 is an example data structure format for registration of configuration parameter IDs;
FIG. 4 is an example format of an ECP file;
FIG. 5 is a flowchart of the functionality of a preferred implementation of an ECP file generator; and
FIG. 6 is a flowchart of the functionality of a preferred implementation of an ECP file reader.

Detailed Description of the Present Invention

[0013] A mechanism for securing software configuration parameters is described herein which provides the ability for a software manufacturer and a customer to customize the software configuration and then to secure the configuration parameters to disallow tampering by unauthorized users. Specifically, after the customer determines its desired customer specific configuration, the configuration parameters and files are secured in an enforced configuration package. At software installation or startup time, the enforced configuration packages are validated to determine whether tampering has occurred. If tampering of the enforced configuration packages is detected, the software application is alerted and handles the tamper message accordingly.

[0014] FIG. 1 illustrates a block diagram of a system in which the invention may operate. In FIG. 1, software application 102 may be configured using any or all of configuration files 112, 114, 116 as determined by the contents of ECP file 106. Software application 102 is typically shipped with all of configuration files 112-116. If no ECP file exists and the software is licensed for a full configuration, application 102 configures itself using all of configuration files 112, 114, 116, ignoring any existing ECP files. If the software is licensed for a limited configuration, however, software application 102 searches for an ECP file. If an ECP file does not exist under a limited configuration license, software application 102 preferably will not allow the user to run it. If an ECP file does exist under a limited configuration license, however, the read ECP file routine 104 reads and validates the ECP file and extracts the customer specific configuration (i.e., the particular configuration files 112 - 116 to utilize when configuring the software). ECP file 106 includes configuration parameter entries which each preferably include a parameter identifier, a param-

eter value, and a parameter fingerprint. ECP file 106 is generated from an ECP description file 108 using an ECP file generator 110. The ECP description file 108 is typically developed by an intermediate customer, or by the software manufacturer working in conjunction with the intermediate customer. The ECP file 106 is then generated at the factory using ECP file generator 110. Preferably, only the ECP file 106 (and not the ECP description file 108) is shipped to end users to prevent end users from tampering with the customer specific configuration parameters.

[0015] As also shown in FIG. 1, application 102 includes an ECP file reader routine 104. In a preferred embodiment, and as embodied in FIG. 1, application 102 is implemented in an object-oriented language such as C++, and the ECP file reader routine 104 is implemented in an ECP object comprising a READ_ECP API method. Application 102 executes a method call on READ_ECP of the ECP object. READ_ECP accesses ECP file 106, validates each entry utilizing its associated fingerprint and any files associated with each entry that are required to also be validated, and either returns the configuration encoded in the ECP file 106 if no tampering is detected, or an error code if tampering is detected. If no tampering is detected, the value returned by the READ_ECP method indicates which configuration files 112-116 the software application 102 is to use for its configuration. For example, if the ECP file includes entries for configuration files 112 and 114, application 102 configures itself using configuration files 112 and 114, but not configuration file 116.

[0016] FIG. 2 is an example format of an ECP description file. As shown in FIG. 2, each configuration parameter entry is set forth on a separate line in the format ⟨ID, value⟩ which includes a configuration parameter identifier (ID) and a configuration parameter value. The configuration parameter ID is a pre-defined identifier whose name and type are known by the ECP file generator 110. The configuration parameter value is the actual value of the configuration parameter. The type of the value must correspond to the type of the ID as defined within the ECP file generator 110.

[0017] Preferably, each configuration parameter ID is a parameter name which is registered within the ECP file generator 110 and which has a parameter type and a set of parameter flags associated with it. The parameter name, type, and flags are preferably defined in a static data structure within the ECP file generator 110. FIG. 3 illustrates an example data structure format for registration of configuration parameter IDs. In the preferred embodiment, a parameter type may be a "string" for elements such as a file name, a "string list" for elements such as a list of event categories or list of files, or an "integer" for elements such as counters or other scalar values.

[0018] In addition, the parameter flags indicate the type of validation that should be performed on the parameter by the ECP file reader routine 104 when the ECP file is validated and read. In the embodiment shown in FIG. 2, the parameter flags may include a "CheckContents" flag which tells the ECP file generator 110 that the configuration parameter, of type string or string list, contains a file or files whose contents must be digitally signed, thus making them tamper proof. The parameter flags may also include a "RelativePath" flag which indicates to the ECP file generator 110 that the configuration parameter is a file whose path is relative from a defined location, which is preferably registered within the ECP file generator 110 in the parameter name data structure. The parameter flags may also include an "Overridable" flag which indicates to the ECP file generator 110 that multiple packages may define the parameter and that at least one of them should have a valid entry. This flag may be used for items such as where the application expects to find either a default item that is always shipped with the software or a customized item that the customer ships to the end user. It will be appreciated by those skilled in the art that the format of the ECP description file may vary from implementation to implementation. Accordingly, the embodiment of FIG. 2 is shown by way of example only, and not limitation.

[0019] In the example ECP description file shown in FIG. 2 and corresponding registration data structures shown in FIG. 3, REG_FILE is a configuration parameter ID of type string list having the CheckContents flag set, which indicates that the fingerprint generated for the configuration parameter should include the contents of the file named in the configuration parameter value, and the RelativePath flag set, to indicate that the named file value may be found at the relative path value "C:\APPLICATION\CONFIG\". In this embodiment, each element in the string list is declared on a separate line but is made available via the READ_ECP API method implemented in the ECP file reader routine 104 as a list of all defined values having ID REG_FILE. As also seen in the example of FIGS. 2 and 3, FILTER_FILE is a configuration parameter ID of type string, and has flags CheckContents, RelativePath, and Overridable set, which indicate that the contents of the named file, FILTER_1 should be included in the fingerprint, that the directory where FILTER_1 may be found is in "C:\APPLICATION\CONFIG", and that the file FILTER_1 may be overridden by a customer specific filter file. As also seen in the example of FIGS. 2 and 3, MAP_NAME is a configuration parameter ID of type string, a USER_COUNT is a configuration parameter of type integer.

[0020] FIG. 4 is an example format of an ECP file generated by ECP file generator 110 from the example ECP file shown in FIG. 2. As shown in FIG. 4, the ECP file looks identical to the ECP description file, except that each configuration parameter entry includes not only a configuration parameter identifier (ID) and a configura-

tion parameter value, but also a digital fingerprint value. In other words, each entry in the ECP file is set forth on a separate line in the format ⟨ID, value, fingerprint⟩. The value of each fingerprint is preferably a digital signature of the configuration parameter ID, value, and contents of the file named in the configuration parameter value if the CheckContents flag is set. Preferably, the digital signature is generated using a combination of a digital signature generation algorithm (e.g., the well-known MD5 algorithm) and an encryption algorithm (e.g., the well-known Tiny Encryption Algorithm (TEA)) to allow the fingerprint to be easily generated, yet difficult to reverse engineer without the proper key. In a preferred embodiment, the proper key for regenerating the fingerprint is encoded into the license registration number of the software application. The purpose of the configuration parameter fingerprint is to ensure that the parameter and its value (and the value's contents, if applicable) have not been tampered with. In addition to each configuration parameter entry having an associated fingerprint, the ECP file embodied in FIG. 4 also includes a beginning of file (BOF) fingerprint and an end of file (EOF) fingerprint. The BOF fingerprint is preferably a digital signature on a customer key assigned to the particular intermediate customer. Typically the customer key is derived from a license registration number shipped with the software. The purpose of the BOF fingerprint is to ensure that the ECP file is used only on the customer's platform and also ensures the integrity of the ECP file and customer key. The EOF fingerprint is a final digital signature on the entire contents of the ECP file. The purpose of the EOF fingerprint is to ensure that the ECP file itself has not been tampered with.

[0021] FIG. 5 is a flowchart of the functionality of a preferred implementation of ECP file generator 110. As shown in FIG. 5, ECP file generator 110 begins with step 502 by initialing its internal data structures, as shown in FIG. 3. Accordingly, a data structure is allocated for each predefined configuration parameter identifier, and the type, flags, and relative path associated with that particular configuration parameter ID are set according to its definition (i.e., ID name, type, flags, and path) as known by the ECP file generator 110. Thus, as illustrated in FIG. 3, a data structure is allocated for each ID (i.e., REG_FILE, FILTER_FILE, MAP_NAME, and USER_COUNT), and the values of its associated type, flags and path are filled in. The configuration parameter ID definitions may be defined statically within the software, or may be configurable as for example by storing all configuration parameter ID definitions in a table or separate file which may be modified to support different configuration parameter IDs.

[0022] In a step 504, the ECP file generator 110 reads the customer key, which may be input as a parameter to the ECP file generator 110 or may be at a location known by the ECP file generator 110. After reading the customer key, the ECP file generator 110 generates a digital signature on the customer key and outputs it as the BOF fingerprint to the ECP file 106.

[0023] The ECP file generator 110 then reads the ECP description file 108 in step 506 to identify ID, value pairs. In a preferred embodiment, the ECP file generator 110 creates a table containing each ID in one column, and all values associated with that ID in a second column.

[0024] In a step 508, the ECP file generator 110 then generates a fingerprint for each of the ID value pairs. The fingerprint for an ID value pair may include the actual name of the configuration parameter ID (e.g., REG_FILE) and the value of the configuration parameter (e.g., CONFIG_1). If the CheckContents flag for the configuration parameter is set, the fingerprint will also include the digital signature of the file that was described by that parameter. As an illustration, the value of each configuration parameter of ID REG_FILE is actually the name of a file. Because CheckContents is set for all configuration parameters having a REG_FILE ID, the contents of each file named as a value of REG_FILE is included in the signature for the ID, value pair.

[0025] In a step 510, the ECP file generator 110 outputs the ⟨ID, value, fingerprint⟩ entries to ECP file 106.

[0026] In a step 512, ECP file generator 110 generates an end-of-file (EOF) fingerprint on the entire contents of the ECP file 106 and appends the EOF fingerprint to the end of the ECP file 106.

[0027] FIG. 6 is a flowchart of the functionality of a preferred implementation of the read ECP file routine 104. As shown in FIG. 6, ECP file routine 104 begins with step 602 by looking for ECP files. In the preferred embodiment, the location of the ECP files are known by the read ECP file routine 104. Preferably the software is shipped with a customer specific default ECP file. To allow intermediate customers to provide different configurations to different end user customers, the intermediate customer may be provided with an ECP file generator to allow it to generate one or more additional ECP files which contain configuration parameters that override the default configuration parameters. This allows the intermediate customer to ship the software manufactured by the software manufacturer containing its customer specific default ECP file along with one or more additional configuration files it may have developed to operate with the software and an accompanying ECP file to prevent end users from modifying the configuration files and parameters. In this embodiment, the ECP file reader routine 104 looks in a particular directory for the default ECP file and any additional customer generated ECP files.

[0028] In a step 604, ECP file reader routine 104 generates a fingerprint on the entire contents of the ECP file 106 and compares it in step 606 to the EOF fingerprint to the end of the ECP file 106. If the fingerprint does not match the EOF fingerprint, the ECP file reader routine 104 returns an appropriate error code in step 618 and exits.

[0029] If the EOF fingerprint is valid, the ECP file generator 110 then reads the customer key in step 608, which may be input as a parameter to the ECP file generator 110 or may be at a location known by the ECP file generator 110. After reading the customer key, the ECP file generator 110 generates a fingerprint on the customer key and compares it to the BOF fingerprint in the ECP file 106 in step 610. If the fingerprint does not match the BOF fingerprint, the ECP file reader routine 104 returns an appropriate error code in step 618 and exits.

[0030] If the BOF fingerprint is valid, the ECP file reader routine 104 then reads the ECP description file 108 in step 612 to identify and match up ID, value pairs. In a preferred embodiment, the ECP file reader routine 104 creates a table containing each ID in one column, and all values associated with that ID in a second column.

[0031] In a step 614, the ECP file reader routine 104 then generates a fingerprint for each ⟨ID, value⟩pair and compares in step 616 the generated fingerprint with the corresponding fingerprint contained in the ECP file. The fingerprint for an ID value pair may include the actual name of the configuration parameter ID (e.g., REG_FILE) and the value of the configuration parameter (e.g., CONFIG_1). If the CheckContents flag for the configuration parameter is set, the fingerprint will also include the digital signature of the file that was named as the value of that parameter. If the Overridable flag is set for an ID, the ECP file reader routine 104 interprets this to mean that more than one possible files exist as the value of this particular configuration parameter. For example, a default filter file may be shipped with the software application which the customer may override with a customer-specific filter file. In this case, it may still be desired to secure the configuration parameter (i.e., the default filter file itself). Accordingly, the CheckContents flag is set to indicate to the ECP file reader routine 104 to include the contents of the default configuration file when generating the fingerprint for the configuration parameter. In addition, the Overridable flag indicates to the ECP file reader routine 104 that a customer specific filter file may exist, and as long as one of the potential configuration files has a valid fingerprint, no error should be signaled.

[0032] Once a fingerprint is generated for an ⟨ID, value⟩pair, the fingerprint is compared to the fingerprint of the ⟨ID, value⟩pair from the ECP file 106 in a step 616. If the fingerprints do not match, an appropriate error is returned in step 618.

[0033] If the fingerprints do match, a step 620 checks for additional ⟨ID, value⟩pairs to validate, and steps 612 - 620 are repeated until each of the ⟨ID, value⟩pairs have been validated.

[0034] In a preferred embodiment of the software application 102 of FIG. 1, which as described earlier is implemented in an object-oriented language such as C++, the ECP object implementing the ECP file reader routine 104 is initialized at startup of the application 102. Accordingly, at initialization, the READ_ECP API method searches for ECP file 106, validates and reads ECP file 106, and returns a list of configuration files (i.e., any combination of configuration files 112 - 116) according to which the application 102 is to be configured. Application 102 then reads each configuration file in the returned list and configures itself according to those configuration files. The listed configuration files describe the look and functionality of the software application to the end user according to the customer's specifications.

[0035] As described in detail above, the present invention provides a method of enforcing a software configuration for end users. In addition, the invention provides a method for ensuring that configuration parameters and configuration files have not been tampered with. It will be appreciated by those skilled in the art that the principles of the present invention may be extended to enforce any type of static mapping of name to values, and to determine whether any type of data storage files have been tampered with. Accordingly, while illustrative and presently preferred embodiments of the invention have been described in detail herein, it is to be understood that the inventive concepts may be otherwise variously embodied and employed and that the appended claims are intended to be construed to include such variations except insofar as limited by the prior art.

Claims

1. A system for securing software configuration parameters used by a software application (102), comprising:

   an enforced configuration packet (ECP) file generator (110) which receives a configuration parameter description file (108) and generates an enforced configuration packet (ECP) file (106), said configuration parameter description file (106) comprising a configuration parameter (⟨ID, VALUE⟩) and said ECP file (106) comprising an enforced configuration packet (⟨ID, VALUE, FINGERPRINT⟩) comprising said configuration parameter and a configuration parameter fingerprint (⟨FINGERPRINT⟩);
   an ECP file reader (104) which validates said enforced configuration packet in said ECP file (106), and returns a return value comprising said configuration parameter if said enforced configuration packet is valid and returns said return value comprising an error code if said configuration parameter is not valid.

2. The system of claim 1, wherein:

   said software application (102) receives said return value from said ECP file reader (104)

and utilizes said configuration parameter if said return value comprises said configuration parameter, and disallows use of said configuration parameter if said return value is an error code.

3. The system of claim 1 or 2, said configuration parameter comprising a file name.

4. The system of claim 3, said configuration parameter fingerprint generated on said configuration parameter and contents of said file name.

5. The system of claim 3, said configuration parameter fingerprint generated on contents of said file name.

6. The system of claim 1, 2, 3, 4 or 5, said ECP file generator (110) generating a key fingerprint (<BOF FINGERPRINT>) on a key parameter (CUSTOMER KEY) and including said key fingerprint in said ECP file (106), said key parameter being required to be known to validate said enforced configuration packet.

7. The system of claim 6, said ECP file reader (104) generating a regenerated key fingerprint on said key parameter (CUSTOMER KEY), comparing said regenerated key fingerprint to said key fingerprint (BOF_FINGERPRINT) in said ECP file (106), and returning an error code if said regenerated key fingerprint and said key fingerprint do not match.

8. The system of claim 1, 2, 3, 4, 5, 6 or 7, said ECP file generator (110) generating an end-of-file (EOF) fingerprint on entire contents of said ECP file (106) after said enforced configuration packets are generated and appending said EOF fingerprint to said contents of said ECP file (106).

9. The system of claim 8, said ECP file reader (104) generating a regenerated end-of-file (EOF) fingerprint on said contents of said ECP file, comparing said regenerated EOF fingerprint to said EOF fingerprint in said ECP file, and returning an error code if said regenerated EOF fingerprint and said EOF fingerprint do not match.

10. The system of claim 1, 2, 3, 4, 5, 6, 7, 8 or 9, wherein:

said configuration parameter ((ID, VALUE)) is overridable, and
said ECP file reader (104) validates a corresponding enforced configuration packet ((ID, VALUE, FINGERPRINT)) in at least one other ECP file, returns a return value comprising said configuration parameter of said enforced configuration packet containing a valid configuration parameter fingerprint, and returns said return value comprising an error code if none of said corresponding enforced configuration packets is valid.

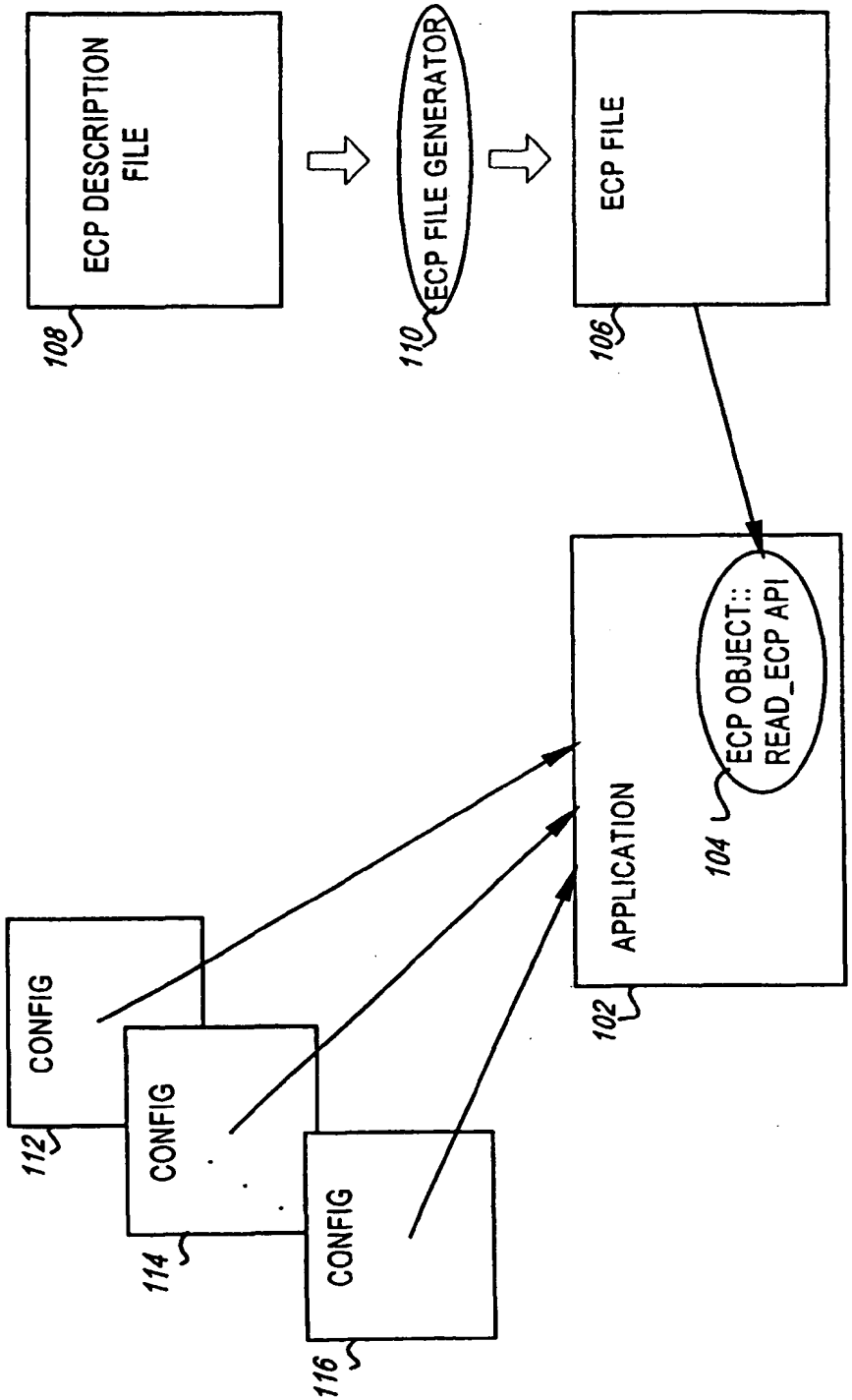

FIG. 1

ECP DESCRIPTION FILE

```
REG_FILE, CONFIG_1
REG_FILE, CONFIG_2
FILTER_FILE, FILTER_1
MAP_NAME, MAP_1
USER_COUNT 1
```

FIG. 2

PARAMETER ID REGISTRATION DATA STRUCTURES

```
REG_FILE:
      TYPE:      STRING LIST;
      FLAGS:     CheckContents, RelativePath;
      PATH:      "C:\APPLICATION\CONFIG\"
FILTER_FILE:
      TYPE:      STRING;
      FLAGS:     CheckContents, RelativePath, Overridable;
      PATH:      "C:\APPLICATION\CONFIG"
MAP_NAME:
      TYPE:      STRING;
      FLAGS:
      PATH:
USER_COUNT:
      TYPE:      INTEGER;
      FLAGS:
      PATH:
```

FIG. 3

ECP FILE

```
<BOF FINGERPRINT>
REG_FILE, CONFIG_1, FINGERPRINT_1;
REG_FILE, CONFIG_2, FINGERPRINT_2;
FILTER_FILE, FILTER_1, FINGERPRINT_3;
MAP_NAME, MAP_1, FINGERPRINT_4;
USER_COUNT, 1, FINGERPRINT_5;
<EOF FINGERPRINT>
```

FIG. 4

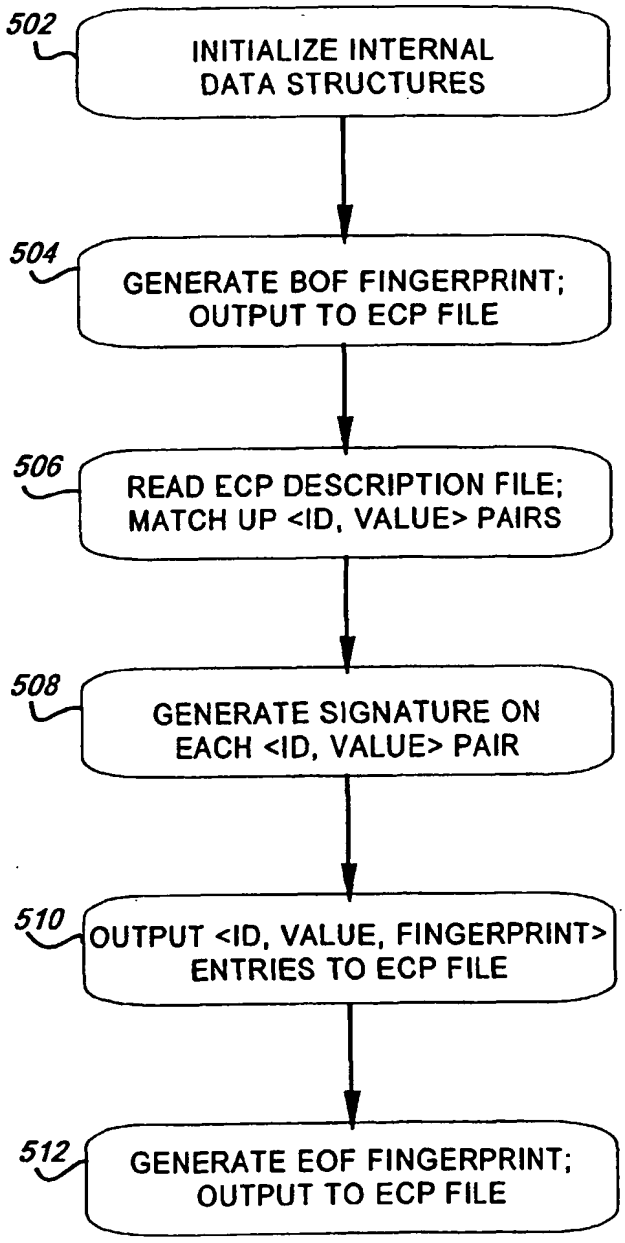

FIG. 5

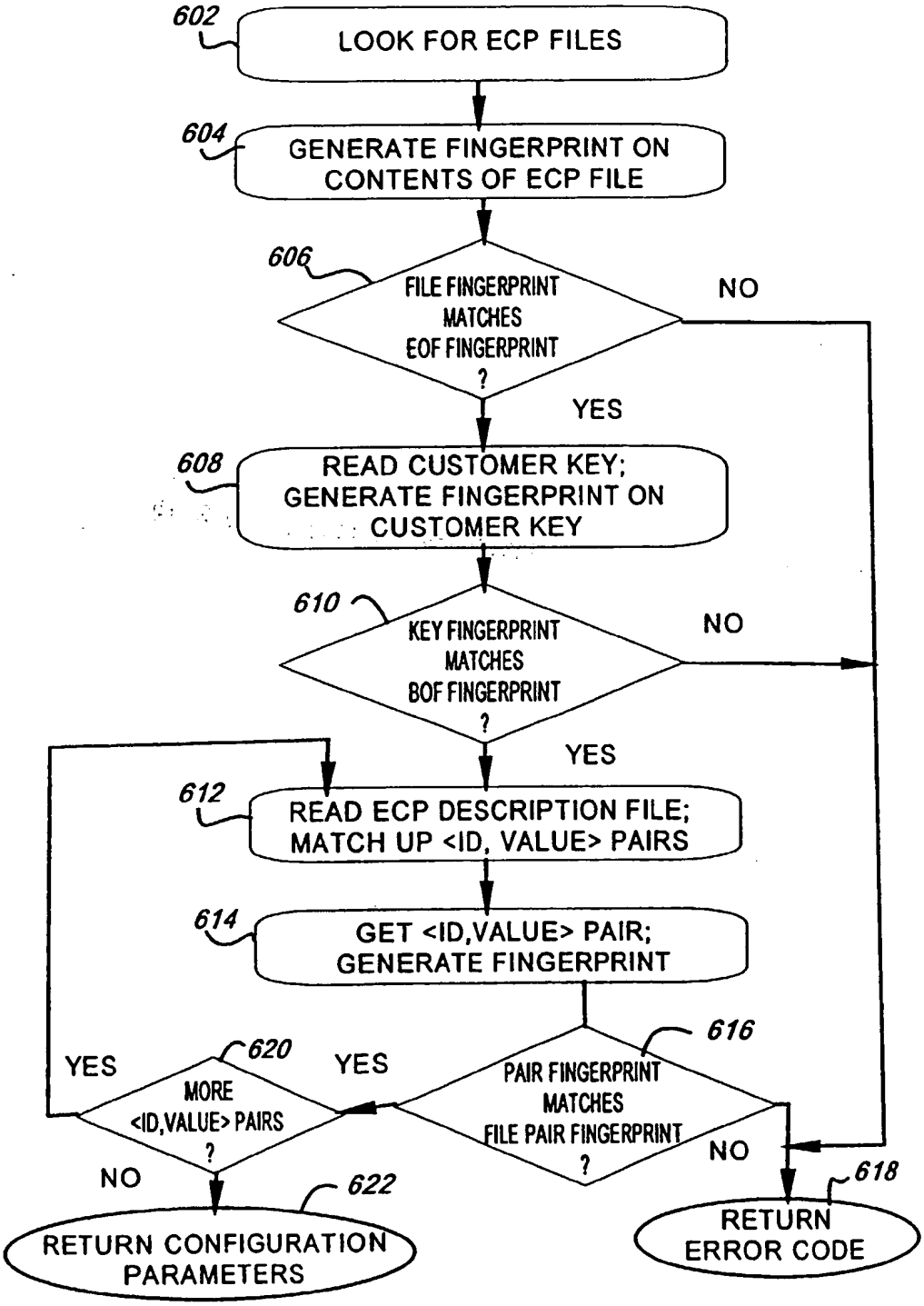

FIG. 6

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 918 275 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
23.02.2000 Bulletin 2000/08

(43) Date of publication A2:
26.05.1999 Bulletin 1999/21

(21) Application number: **98111462.2**

(22) Date of filing: **22.06.1998**

(51) Int. Cl.[7]: **G06F 1/00**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **13.11.1997 US 970064**

(71) Applicant:
Hewlett-Packard Company
Palo Alto, California 94304 (US)

(72) Inventors:
- **McCollom, William Girard**
  Fort Collins, CO 80525 (US)
- **Garcia, Julio Cesar**
  Fort Collins, CO 80525 (US)
- **Smith, Darren Drew**
  Fort Collins, CO 80526 (US)

(74) Representative:
Schoppe, Fritz, Dipl.-Ing.
Schoppe, Zimmermann & Stöckeler
Patentanwälte
Postfach 71 08 67
81458 München (DE)

(54) A method of securing software configuration parameters with digital signatures

(57) A system and method for enforcing configuration parameters and detecting tampering of configuration files used by a software application. An enforced configuration packet (ECP) file generator (110) generates an enforced configuration packet (ECP) file (106) from a configuration parameter description file (108) containing a set of configuration parameters (ID, VALUE). The ECP file (106) includes a set of enforced configuration packets (ID, VALUE, FINGERPRINT), which each include one of the configuration parameters from the ECP description file (108) and a corresponding configuration parameter fingerprint (FINGERPRINT) unique to that particular configuration parameter. At startup of the software application (102), an ECP file reader (104) validates the ECP file (106) and each of the enforced configuration packets contained in the ECP file. Validation is achieved by regenerating the configuration parameter fingerprint of each configuration parameter and comparing the regenerated fingerprint to the fingerprint contained in the enforced configuration packet. If any of the fingerprints in the enforced configuration packets do not match their regenerated fingerprint, the ECP file reader (104) indicates that the configuration parameter fingerprint is not valid. If all of the configuration parameter fingerprints match up to their regenerated fingerprints, and the ECP file (106) itself is determined to be valid, the configuration parameters encoded in the ECP file (106) are used by the software application (102) to set up its configuration.

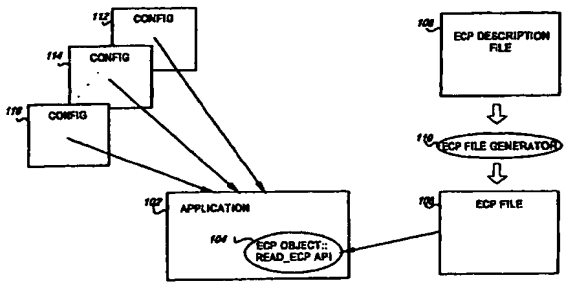

FIG. 1

EP 0 918 275 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 98 11 1462

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	WO 90 13084 A (EMPIRICAL RESEARCH SYSTEMS INC) 1 November 1990 (1990-11-01) * page 14, line 1 - line 26; claims 3,5-8 *	1-5	G06F1/00
E	WO 98 36517 A (JPC INC.) 20 August 1998 (1998-08-20) * page 28, line 1 - page 32, line 10 *	1-3,8,9	
A	WO 93 02419 A (J.A.S. TECHNOLOGY (AUSTRALIA) PTY. LTD) 4 February 1993 (1993-02-04) * page 5, line 22 - page 6, line 16 *	1	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	21 December 1999	Taylor, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 98 11 1462

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

21-12-1999

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9013084 A	01-11-1990	US 5144659 A	01-09-1992
		AU 5448390 A	16-11-1990
		CA 2014868 A	19-10-1990
		EP 0422184 A	17-04-1991
		US 5289540 A	22-02-1994
WO 9836517 A	20-08-1998	US 5953502 A	14-09-1999
WO 9302419 A	04-02-1993	AU 2366292 A	23-02-1993

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82